**IFN 563: Object Oriented Design**

Assessment 2: Final design and implementation

Taiga Ishihara- N11233940

Email Address : [d-taiga.4869@outlook.jp](mailto:d-taiga.4869@outlook.jp)

Carl Profke- N11525304

Email Address : [n11525304@qut.edu.au](mailto:c.schmitz@qut.edu.au)

Ikufumi Mitsuyu-N11535857

Email Address : [ikufumimitsuyu@icloud.com](mailto:ikufumimitsuyu@icloud.com)

**1 Statement of Completion**

Although there are some features that we have not been able to include in our games, we have successfully implemented both games so that players can enjoy both games from the start to the end.

Specifically, players can choose a game mode for both games between human vs human and human vs computer. Next, when they choose moves, it is automatically checked if the moves are valid or not. For playing with computer, it generates random numbers which makes random moves for both games.

However, we have not been able to implement some features that would make the game more fascinating. For example, we abandon on Undo function, Load function and Assistance function. Also, we have made the save function work, but players cannot save and play the game whenever they want to during the game, so the game data is stored at the end of the game in the gamesData.json.

**2 Overview of Final Design**

One of the most noticeable aspects of our design is the Template Method pattern. This method compresses a significant amount of game code into a single function within the Game class beneath the main method. This approach enables us to keep the main method very concise.

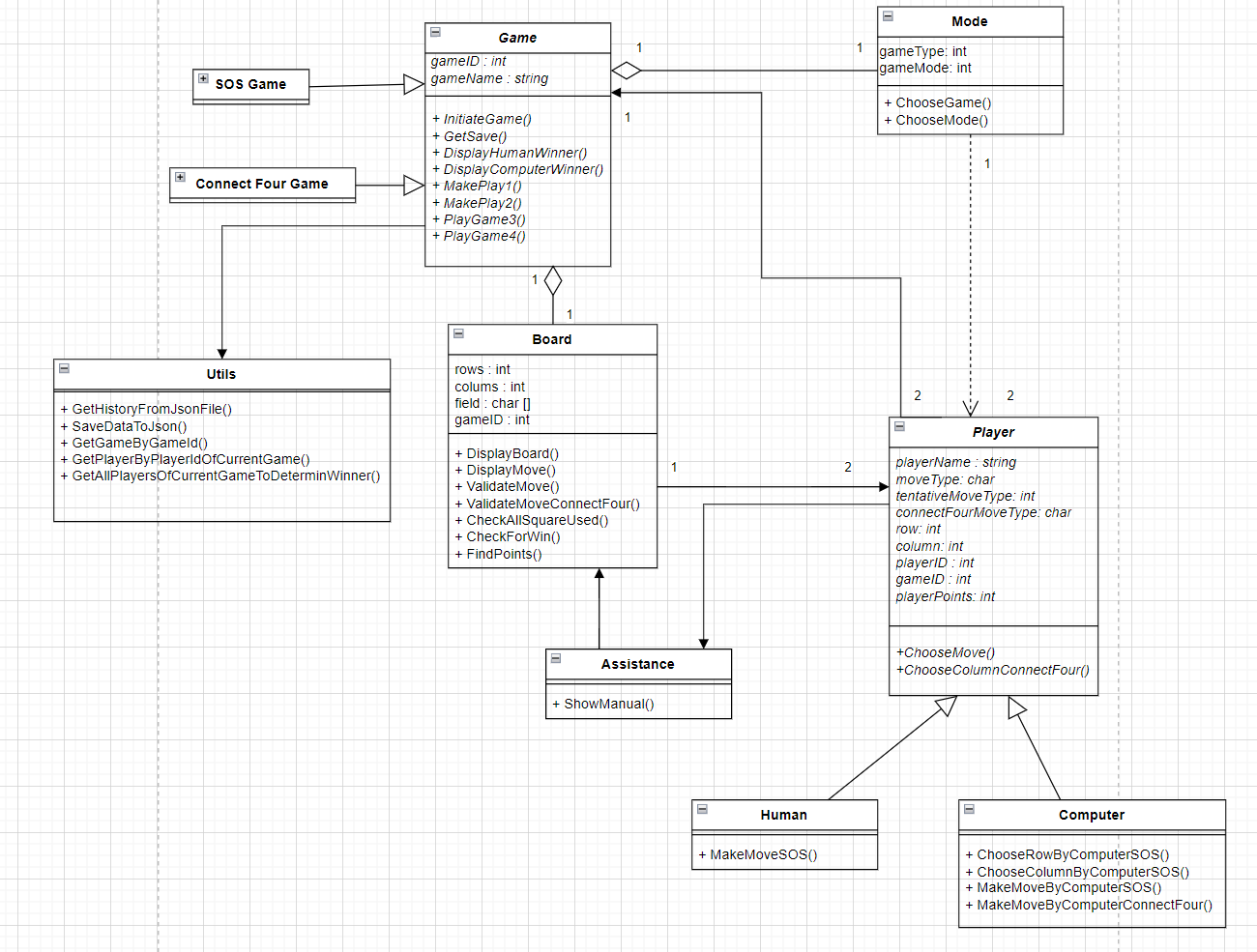
When it comes to changes from the previous design, the History class has been removed because C# console always shows us every move history with a board.

The Save class was removed and in exchange we added Utils class which included several new methods so that the game data can be saved properly, using json file. The Utils class provides a set of utility methods that can be used to perform synchronization operations and create new objects. By using the Utils class, it cleaned re-occuring code and removed complexity. As it is always static, it doesn't have to be initialized.

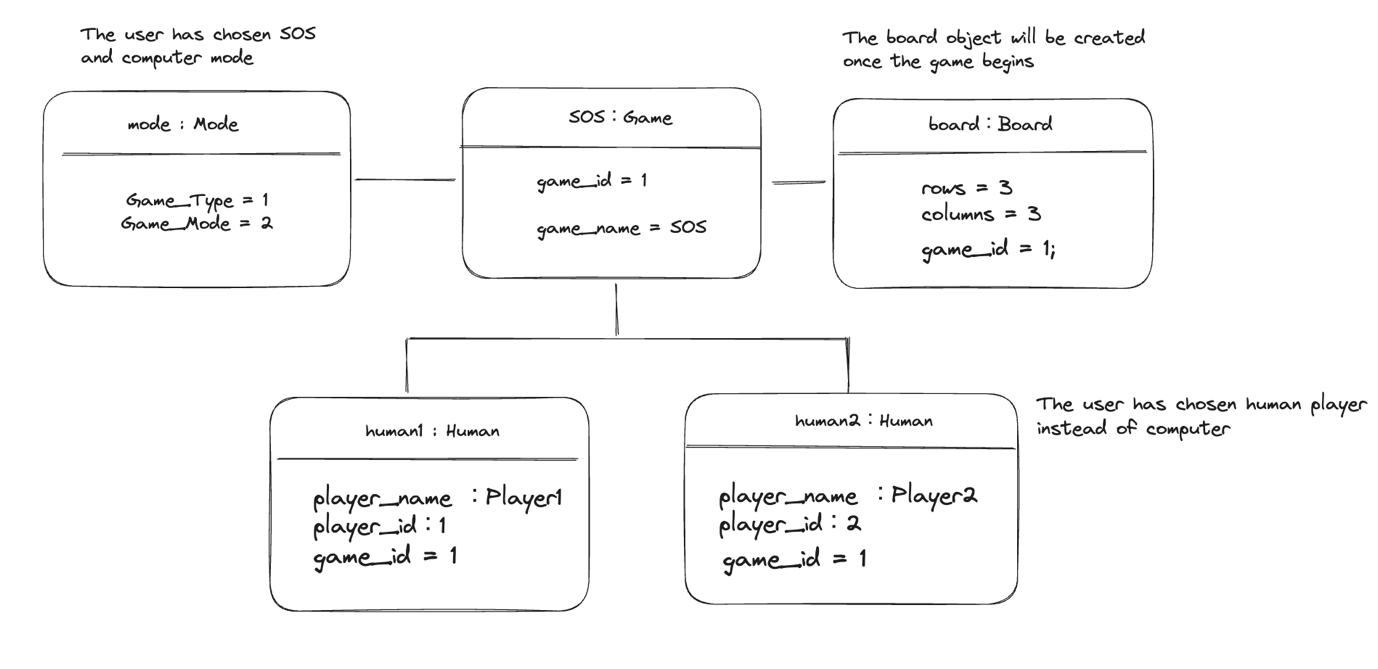
Another significant change is the differentiation between methods for human players and computer players. While they share the same method names, the method for human players prompts user input, whereas the one for computers generates random moves that are then converted into move types, such as 'S' or 'O'.

**3 Detailed Final Design Documents**

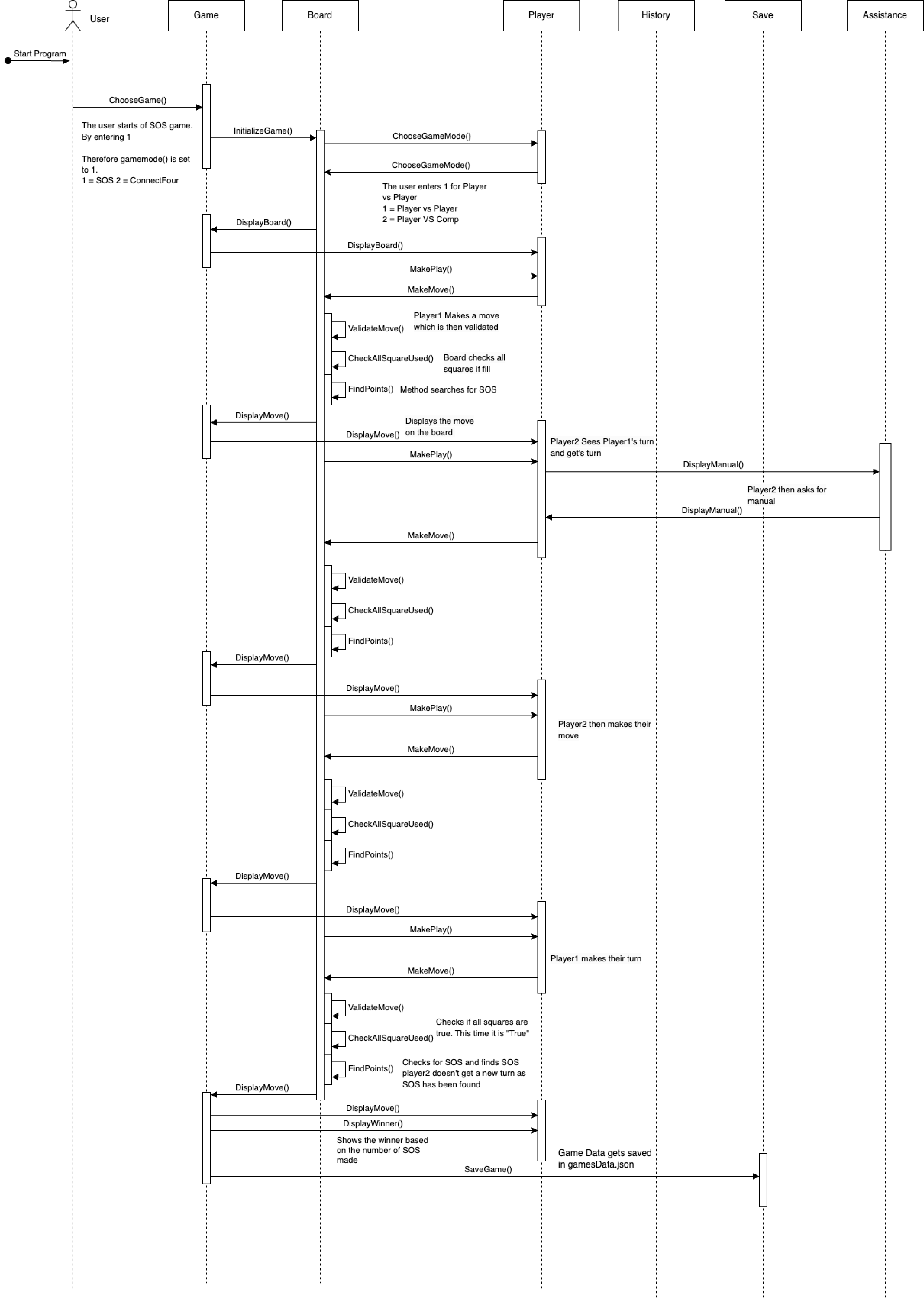
1) Class Diagram



2) Object Diagram



**3 Sequence Diagram**

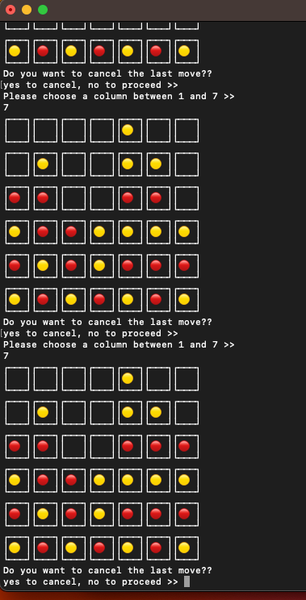


**4 Justification of Design Principles and Patterns**

We used Template method (Player Game1 ~ 4) in Game class. The important operations in the design pattern would include InitiateGame method, MakePlay1, MakePlay2, DisplayHumanWinner, and DisplayComputerWinner. The design pattern has made the main class much more concise because the Template method contains almost all operations for the game from the start to the end and we just needed to place the method in the main class.

5 How our program can be executed

The program can be executed in Visual Studio or any other IDE which is capable with C#, we would recommend running it in Visual Studio. Open the project in IFN563\_Assignment. You can do this by selecting "File" -> "Open" -> "Project/Solution" and navigating to the location of the project file. Before running the program, you need to also make sure of the rest of the files are there including the GamesModel.cs as it is a requirement to run the code. Another recommendation would be to simply run the program on your terminal. Make sure that you have the .NET SDK installed on your system. You can check by running dotnet --version. Navigate to the directory with cd and compile and run it using this command “dotnet run Program.cs” The program should run on the terminal with this method, a benefit of using the terminal is that Unicode works. I have provided photos of the game using Unicode. This makes a better-looking game as the terminal can display Unicode. However, we were not sure if Unicode would work in all situations, so we decided to use characters as Visual studio would not let us run the program with the Unicode symbols.



Reference photo of the Game running with Unicode(Not in the final version, draft phase)

6. Classes/interfaces from existing libraries and frameworks.

We have not imported any libraries in our project.

7. Contributions

In our team, every member made valuable and equitable contributions to the assignment, ensuring our best to successfully complete the assignment. Each team member brought unique skills and perspectives to the table. All members played a crucial role in designing the program's architecture and implementing core functionalities. All members focused on user interactions, enhancing the user interface, and ensuring a smooth gaming experience and contributed to each other by thoroughly testing the program, identifying and fixing bugs, and improving overall program stability. All collaborated in creating the three the diagrams to provide a visual representation of the program's structure, including class diagram, sequence diagram and object diagram. Effective communication and collaboration were key throughout the project, with everyone actively participating in discussions, sharing ideas, and providing constructive feedback. This equal distribution of tasks and responsibilities allowed us to create a well-rounded and functional program while fostering a sense of teamwork and unity within the group.